What is claimed is:

1. An apparatus for designing a real-time system involving a plurality of tasks, the apparatus comprising:

an asynchronous factor tester testing whether or not task operations arranged in time series in each section of the real-time system are affected by asynchronously occurring factors.

2. The apparatus of claim 1, wherein the asynchronous factor tester comprises:

a parameter extractor extracting the asynchronously occurring factor of each section of the real-time system; and

a branch tester testing whether or not the asynchronously occurring factor occurs branching of the each section.

3. The apparatus of claim 1, wherein the asynchronously occurring factor includes interrupt and time-out process.

4. An apparatus for designing a real-time system involving a plurality of tasks, the apparatus comprising:

a connector connecting sections each containing a series of tasks that are not affected by asynchronously occurring factors, to one another to form the real-time system; and

a tester testing if the connected sections operate consistently.

5. The apparatus of claim 4, wherein the tester comprises:

a reader reading a task start state of the rear section of the real-time system and a task end state of the front section of the real-time system;

a comparator comparing the task start state of the rear section with corresponding the task end state of the front section; and

a determination device determining whether or not the task start state of the rear section agree with the task end state of the front section.

6. The apparatus of claim 5, wherein the task state includes Dormant, Wait, Ready and Run.

7. A method for designing a real-time system involving a plurality of tasks, the method comprising:

testing whether or not task operations arranged in time series in each section of the real-time system are affected by asynchronously occurring factors.

8. The method of claim 7, wherein the testing operation comprises:

extracting the asynchronously occurring factor of each section of the real-time system, and

testing whether or not the asynchronously occurring factor occurs branching of the each section.

9. The method of claim 7, wherein the asynchronously occurring factor includes interrupt and time-out process.

10. A method for designing a real-time system involving a plurality of tasks, the method comprising:

connecting sections each containing a series of tasks that are not affected by asynchronously occurring factors, to one another to form the real-time system; and

testing if the connected sections operate consistently.

11. The method of claim 10, wherein the testing operation comprises:

reading a task start state of the rear section of the real-time system and a task end state of the front section of the real-time system;

comparing the task start state of the rear section with corresponding the task end state of the front section; and

determining whether or not the task start state of the rear section agree with the task end state of the front section.

12. The method of claim 11, wherein the task state includes Dormant, Wait,

Ready and Run.

13. A computer program product for designing a real-time system involving a plurality of tasks, the computer program product comprising:

testing whether or not task operations arranged in time series in each section of the real-time system are affected by asynchronously occurring factors.

14. The computer program product of claim 13, wherein the testing operation comprises:

extracting the asynchronously occurring factor of each section of the real-time system; and

testing whether or not the asynchronously occurring factor occurs branching of the each section.

15. The computer program product of claim 13, wherein the asynchronously occurring factor includes interrupt and time-out process.

16. A computer program product for designing a real-time system involving a plurality of tasks, the computer program product comprising:

connecting sections each containing a series of tasks that are not affected by asynchronously occurring factors, to one another to form the real-time system; and

testing if the connected sections operate consistently.

17. The program of claim 16, wherein the testing operation comprises:

reading a task start state of the rear section of the real-time system and a task end state of the front section of the real-time system;

comparing the task start state of the rear section with corresponding the task end state of the front section; and

determining whether or not the task start state of the rear section agree with the task end state of the front section.

18. The program of claim 17, wherein the task state includes Dormant, Wait,

Ready and Run.